Leading Computational Methods on Ultra-Scale HEC Platforms

1 Project Summary

Computational scientists have seen a frustrating trend of stagnating application performance despite dramatic increases in the claimed peak capability of high performance computing systems. This trend has been often attributed to one or more of the following factors: the gap between memory and arithmetic performance, the reliance on cache-based memory hierarchies that are a poorly matched for certain algorithms, the lack of integration between the networks and processors due to the use of commodity technology, and the use of low degree interconnect topologies. In general, the machines are not well balanced for the demands of large-scale numerical computations.

The problem is expected to be exacerbated by the end of this decade, because traditional uniprocessor scaling trends are slowing down, as most of the benefits of instruction level parallelism and pipelining have been mined, and clock frequency increases are limited by power dissipation concerns. In the same time frame, mission-critical applications will have computational requirements that are at least two orders of magnitude larger than current levels [16, 17, 59]. Several different approaches have been developed to address these issues, including the use of parallel vector systems, ultra-scale system built from low-power components, scalable shared address space architectures, and systems with multiple networks designed for different performance criteria. Over the next few years, more system architectural innovations are likely both in commodity components, e.g., single chip multiprocessors, and from research programs like DARPA's HPCS effort.

The goal of the proposed work is to evaluate existing and emerging large scale HEC architectures using a set of in-depth studies from full applications. The novel aspect of this research is the emphasis on full applications, run with real input data and at the scale desired by application scientists in the domain. These problems are much more complicated than in traditional benchmarking suites such as the NAS Parallel Benchmarks or the Linpack benchmark, and therefore reveal the kinds of performance issues that arise in applications that combine multiple physical models, perform large volumes of I/O, and for which the underlying kernels can be organized to in various ways to match the features of a given architecture. The research is also much more difficult to perform, because it requires coordination, communication, and management of a team of application scientists in highly disparate backgrounds. The detailed application analysis reveals algorithmic limitations to scalability and cases where a particular code is poorly suited to an architectural features such as vectors, caches, or loosely integrated networks. The head-to-head comparison of applications across architectures, combined with a careful analysis of the applications characteristics, provides invaluable insight into the suitability of a given type of machine for a particular class of scientific methods.

The high level goal of our work is to provide insight into the effectiveness of various architectural models for HEC with respect to absolute performance and scalability across a broad range of DOE Science application domains. Our head-to-head comparisons on full applications help in identifying the suitability of a particular architecture for a given service site or set of users, gives application scientists and applied mathematicians information about how well various numerical methods perform across systems, and reveal performance-limiting system bottlenecks that can aid designers of the next generation systems. The in-depth studies reveal limitation of compilers, operating sytsems, and hardware, since all of these components must work together at scale to achieve high performance. Overall, we believe these results will lead to more efficient use of the community resources in both current installation and in future designs.

2 Key Research Questions

As of a decade ago, the trend in HEC systems was clearly towards building clusters from commodity components. The traditional model is to use the high performance processing nodes, typically shared memory multiprocessors (SMP), as the computational building blocks. Today one sees a much more diverse set of HEC models, due in part to the low efficiency numbers observed on commodity clusters and the related re-emergence of vector computing led by the successful Earth Simulator System. The SGI Altix system takes the cluster of SMP model to the extreme by scaling the shared memory nodes to 512 processors using distributed shared memory and providing a tightly integrated network that offers low latency communication through a global address space model, even beyond 512 processors. At the opposite extreme, the desire for scalability combined with concerns over power consumption has led to the introduction of ultra-scale parallel machines such as the BG/L, which use relatively low end processing components in huge numbers. At the same time, the vector parallel model is still prominent in machines like the Cray X1 and NEC SX8. The landscape of architectures is complicated because each architecture, or even each system instance, constitutes a unique combination of processing, memory system, and networking features. Price performance still remains a goal of most computing centers, which makes careful performance comparisons across architectures and application domains essential.

The proposed work will follow the successful model used in our prior DOE project "Modern Vector Evaluation," where we worked closely with application scientists to: identify a broad set of key applications, investigate performance using realistic simulation parameters and input sets; port full-scale applications onto leading HEC architectures; optimize complex algorithms to match the features of these platforms; and disseminate our results to the HPC community. The prior work was focused on vectorization and questions related to the effectiveness of vector machines to meet scientific demands. In the proposed work we will address the effectiveness of ultra-scale architectures and shift the emphasis from vectorization to network degree and level of integration. We will also continue looking at vectorization, but will extend the study to include new applications domains, such as adaptive mesh refinement, which have different parallelism models than in our previous work. Our study will examine several research questions related the current spectrum of HEC systems:

1. How efficient are ultra-scale, low-power machines for DOE Science applications either for a particular class of numerical methods, a particular application domain, or across a broad set of applications?
2. Given the cost of high degree (e.g., fat-tree) networks, especially as the number of processors scale, under what circumstances can low-degree interconnect networks be used effectively?
3. Does tight network integration, either for providing low latency MPI or shared address space programming model, provide a significant benefit to some applications?
4. Given the limitations of single processor scaling, what types of fine grained (on-chip) parallelism will be most effective across applications? In particular, does vector processing provide the most important type of fine-grained parallelism for scientific applications?
5. Modern vector machines and shared memory machines provide a wide range of memory system architectures, including cacheless, cache-coherent, or cached without coherence. How do these features affect application performance?
6. What is the value of shared memory hardware (e.g., the Altix CC-NUMA model in high end systems) and how does the resulting architecture compare in a very large-scale system to a flatter, ultra-scale, model or to a less sophisticated commodity-based interconnect interfaces that do not support coherence.

This initial set of questions may expand as the landscape of available HEC architectures changes, for example, to accommodate large machines built with single chip multiprocessors, novel memory systems, or other features.

3 Relation to Prior Work

Several other groups have done performance comparisons across networks, but we believe that the breadth of our application suite combined with the depth of each individual application study, which often involve significant performance tuning, makes our work unique.

3.1 Modern Vector Evaluation

Our prior work resulted in several technical insights into the benefits and limitations of vector parallel machines relative to more conventional superscalar-based machines [18, 23, 25, 47, 48, 52, 53, 54, 58]. Our research team was the first international group to conduct a performance evaluation study at the Earth Simulator (ES) Center (remote ES access in not available). Additionally, our preliminary study has been recognized as an FY2004 accomplishment in the FY2006 congressional budget [1]. This type of work is also highly beneficial to numerous government activities pursuing HEC productivity, including the DOD High Performance Computing Modernization Program (HPCMP) [3], and the High End Computing University Research Activity (HEC-URA) [5]. We have directly contributed to the DARPA High Productivity Computing Systems (HPCS) [4] program, by working closely with David Koester (Productivity Team Leader) and providing two full-scale application codes that have been integrated into their benchmark suite.

We will build on the proven track record of close collaborations with the HPCS community to help ensure that state-of-the-art computational methods will map well onto future peta-flop system designs. Our group is part of the Berkeley Institute for Performance Studies, which brings together researchers at LBNL and UC Berkeley to work on all aspect of performance analysis, tuning, and modeling. Our group also works closely with application scientists throughout DOE and with the NASA Advanced Supercomputing Division (NAS) Division.

The proposed work differs from that in our previous study by looking in detail at network characteristics, ultra-scale systems, and innovations in vector memory systems. We will also expand the suite of evaluated codes to cover new algorithmic techniques and thereby cover most of the space of computational methods. Finally, our new effort will focus on the communication requirements of these high-end applications, to help network designers and system procures prepare for next-generation ultra-parallel computing platforms.

3.2 Related Work

The purpose of any HPC system is to run full-scale applications, and performance on such applications is the final arbiter of the utility of the system — thus comparative application performance data must be readily available to the HPC community at large. However, evaluating large-scale scientific applications using realistic problem sizes on leading HEC platforms is an extremely complex process, explaining why relatively few direct application comparisons are currently available across architectures. The literature does include a number of more narrowly focussed studies of a particular architecture, such parallel vector machines [18, 31, 33, 45, 55, 65].

One approach to reducing the complexity of architectural evaluation is the development of better benchmarking methods to improve program characterization and performance prediction. Scalable Compact Application (SCA) benchmarks are derivatives that capture major characteristics of the full applications, but avoid many of their idiosyncrasies by leaving out unnecessary details. The most popular example in this category, for scientific computations, are the NAS Parallel Benchmarks (NPB) [6]. Another ongoing effort to is develop tunable synthetic benchmarks. These parameterized codes enable the researcher to uncover peaks and valleys in a continuum of performance characteristics. The Application Performance Characterization (Apex-Map) [61] is one such effort. The preliminary focus of Apex-Map is to define performance characteristics of application data access patterns and to create a corresponding memory access probe benchmark. These benchmarking efforts are an important step in understanding the mapping between algorithms and architectural features, and are especially usedulf in identifying particular hardware or compiler features that limit performance. However, they rarely mimic the behavior of full-scale applications which may be composed of multiple code phases, each with widely varying requirements.

Another approach to understanding and predicting application performance on various architectural platforms, is through performance modeling. The goal of performance modeling is to gain understanding of computer system's performance on various applications, by means of measurement and analysis, and to encapsulate these characteristics in a derived model. A number of important research efforts are developing these modeling methodologies including, including the Performance Evaluation Research Center (PERC) [13], which includes the Performance Modeling and Characterization (PMaC) group, and the Performance and Architecture Laboratory (PAL) at Los Alamos National Laboratories [9]. Performance modeling is an important component in understanding performance on current and, especially, future machines, but it does not replace the need for detailed application performance analysis. Model development is time consuming and specific to each application, and there is a trade-off between the accuracy of the model and its simplicity, with measured performance providing the data necessary

for validation of the models.

Our work complements these efforts, providing benchmark designers and performance modeling teams with a consistent baseline of performance data for realistic scientific applications on leading HEC platforms. It will help identify smaller benchmarks and parameter settings of interest to our larger applications, and provide the data necessary for validation. Additionally, our analysis of performance-limiting architectural features allows hardware designers, compiler writers, and OS developers to gain a better understanding of scientific application requirements — information that is scarcely available today.

3.3 Coordination with DOE-Supported Projects

Our study also complements other DOE-supported projects relating to parallel performance modeling, prediction, benchmarking, and tools. We plan to work closely with the SciDAC Performance Evaluation Research Center (PERC) [13], headed by David Bailey at Lawrence Berkeley National Laboratory (LBNL). We will benefit from PERC's benchmarking experience, tools, and instrumented codes, while providing candidate applications and performance characterization for their ongoing studies. We will also coordinate our benchmarking activities with Patrick Worley, leader of the Evaluation of Early Systems [7] project at Oak Ridge National Laboratory (ORNL). Our applications are also of interest to computer systems research efforts. For example, we having recently established a cooperation with Dr. Patricia Teller [11], a PI of the FastOS project, to support their operating systems research effort. Additionally, ORNL is currently acquiring a leadership-class system, called Rainier, consisting of both vector and scalar building blocks. We will coordinate our efforts with the ORNL evaluation team to ensure that our benchmarking projects are complementary, and to help identify the algorithmic classes best suited for the Rainier infrastructure.

4 Architectural Research Platforms

Here we briefly describe the unique features of emerging architectural platforms that will be examined in our study, in the context of full-scale scientific applications. Over the past several years a variety of architectural configurations have emerged, including: clusters of workstations, superclusters (constellations) of cache-based symmetric multiprocessors (SMP), massive clusters of embedded low-power processors, and clusters of vector SMPs.

One important trend in HPC computing, is the vast degree of parallelism that will soon be required. As the field of scientific computing matures, the demands for computational resources are growing at a rapid rate. It is estimated that by the end of this decade, numerous mission-critical applications will have computational requirements that are at least two orders of magnitude larger than current levels [16, 17, 59]. However, as the pace of processor clock rate improvements continues to slow, the path towards realizing peta-scale computing is increasingly dependent on scaling up the number of processors to unprecedented levels and/or utilizing a high-degree of on-chip, fine-grained parallelism. The tradeoffs between these approaches will be a central theme in our investigation.

4.1 Ultra-Scale, Low Power Architectures

The most common approach to building supercomputers over the past decade has been to cluster multiprocessor nodes together using high performance interconnect switches. However these machines, at large scale, consume an enormous amount of power and require substantial facilities. The IBM Blue Gene/L (BG/L) system [29] represents a departure from these approaches — allowing unprecedented levels of parallelism, with a low unit cost and low power consumption characteristics. It combines a very large number of low-power processors together with multiple interconnection networks. The BG/L DOE/NNSA system located at LLNL was first powered-on in June 2003 and is the most powerful supercomputer in the world [15], containing 32K nodes with a sustained LINPACK performance of over 70 Tflop/s (the recent upgrade of this system attained over 135 Tflop/s on the LINPACK benchmark) However, several tradeoffs were necessary to achieving this design. Unlike the fully-interconnect networks (crossbar, fat-tree) of many leading supercomputers, BG/L's main communication interconnect is a 3D torus. Additionally, each BG/L node contains relatively modest PowerPC 440 processors, capable of 5.6Gflop/s peak and

support for only 512MB of main memory. Our study will evaluate this unique platform in the context of leading vector systems, and identify the tradeoffs between these two dramatically different architectural approaches.

4.2 Low-Degree Network Connectivity

HEC systems implementing *fully-connected networks* (FCNs) such as fat-trees and crossbars have proven popular due to their excellent bisection bandwidth and ease of application mapping for arbitrary communication topologies. In fact, as of November 2004, 94 of the 100 systems in the Top500 [15] employ FCNs (92 of which are fat-trees). However, it is becoming increasingly difficult and expensive to maintain these types of interconnects, since the cost of an FCN infrastructure composed of packet switches grows superlinearly with the number of nodes in the system. Thus, as supercomputing systems with tens or even hundreds of thousands of processors begin to emerge, FCNs will quickly become infeasibly expensive. This has caused a renewed interest in networks with a lower topological degree, such as mesh and torus interconnects (like those used in IBM BlueGene/L, Cray RedStorm, and Cray X1), whose costs rise linearly with system scale. However, the subset of scientific computations with communications patterns that can be effectively embedded onto these types of networks is yet to be determined by the HPC community.

Several studies that profile application communication have observed that many applications have communication topology requirements that are far less than the total connectivity provided by FCN networks. For instance, the most demanding applications investigated by Vetter and Mueller [63, 64] indicate that the applications that scale most efficiently to large numbers of processors tend to depend on point-to-point communications patterns where each processor's average *topological degree of communication* (TDC) is a relatively few distinct destinations, or neighbors. This provides evidence that many application communication topologies exercise a small fraction of the resources provided by FCN networks.

In order to understand the feasibility of utilizing low-connectively network solutions, interconnect architects and system procurers must first understand the communication requirements of full-scale scientific applications. As part of our study, we plan develop a methodology for addressing this question, and perform an extensive study on our suite of scientific codes.

4.3 Effectiveness of Commodity Platforms

Over the last decade, the architectural spectrum of large-scale parallel systems has been dominated by the cluster-of-workstation paradigm. These systems are characterized by loose hardware integration between the networking and the CPU/memory components (PCI buses), and lack the efficiency of the custom designed computing platforms. As a result, while processor performance evolves exponentially according to Moore's Law, improvements in communication latency, communication software overhead, and sustained performance lag far behind. Numerous developments in commodity system design have the potential to address these deficiencies. Examples include significantly increased memory bandwidth; on-chip memory controllers; and open industry standard, high-performance network technology (InfiniBand). Our study will explore the tradeoffs between utilizing low-cost commodity-based components and custom, tightly-integrated vector systems, for a broad spectrum of scientific applications.

We now briefly describe the commodity-based systems that will be examined in our evaluation.

4.3.1 IBM Power-based Architectures

The IBM Power series is one of the most popular building blocks for supercomputing platforms, and we plan to examine this technology in great detail. We will continue our performance study of Seaborg, the 6080-processor IBM Power3 (1.5 Gflop/s peak) system located at LBNL, and connected via the custom Colony switch. Our evaluation will also examine Cheetah, the 864-processor Power4 (6 Gflop/s peak) system located at ORNL, and interconnected via the custom Federation switch. Additionally, we look forward to evaluating the Power5 system — especially as configured for ASCI Purple. The Power5 promises a number of significant improvements compared with its predecessors that may allow higher sustained performance on scientific applications, including: improvement in memory bandwidth (in absolute terms and as a ratio of peak performance),huge 36 MB L3 cache

on the MCM operating at 50% of CPU speed (compared with 33% of the Power4), on-die memory controller supporting DDR2-SDRAM, improved prefetching support, and a tighter integration to the Federation interconnect. We also look forward to studying the Power6 when it become available.

4.3.2 Intel Itanium-based Architectures

Recently the Intel Itanium2 has become utilized as a building-block for high-performance computing systems. We will explore performance of Lawrence Livermore National Laboratory's (LLNL) Thunder system, currently the fifth most powerful supercomputer [15] in the world. Thunder is a highly-integrated computing system, comprised of 1024 nodes of 4-CPU Itanium2 processors, and interconnect via the Quadrics QsNet ELan4 network. (Preliminary SX-8 experiments have recently been conducted [54]). The Itanium2 cannot store FP data in L1 cache (only in L2), making register loads and spills a potential source of bottlenecks; however, the relatively large FP register set helps mitigate this issue. The superscalar Itanium2 processor performs a combination of in-order and out-of-order instruction execution referred to as Explicitly Parallel Instruction Computing (EPIC). Instructions are organized into VLIW bundles, where all instructions within a bundle can be executed in parallel. However, the bundles themselves must be processed in order. We will also continue our experiments on Ram, the 256-Itanium2 Altix 3000 located at ORNL. Additionally we will explore performance of the Itanium2-based of the NASA Columbia system as described in section 4.6. Finally, we plan to investigate future generations of Itanium-based processors as they become available.

4.3.3 AMD Opteron-Based Architectures

Finally, we plan to evaluate the performance of supercomputing platforms utilizing AMD Opteron processors. The primary floating point horsepower of the AMD Opteron comes from its SIMD floating-point unit accessed via the SSE2 instruction set extensions. The Opteron processor's on-chip, 1 MB L2 cache supports aggressive out-of-order execution and can issue up to nine instructions simultaneously. The integrated memory controller eliminates the need for a separate memory controller chip, providing a low latency path to local memory. The Opteron's HyperTransport technology enables tight integration between the processor and switch fabric, removing the PCI bottleneck inherent in many clustered architectures.We are particularly interested in exploring the performance of the 10,000-processor RedStorm system, soon to be available at Sandia National Laboratories, as well as the leadership class Cray XT3 supercomputer to be installed at ORNL. Additionally, we will examine Jacquard, a 640-processor Opteron system interconnect with InfiniBand technology (currently in acceptance testing at LBNL), to evaluate the tradeoffs between the commodity InfiniBand solution and the Cray custom interconnect design. We also plan to investigate future generations of AMD 64-bit processors, as they become available.

4.4 Fine-Grained, On-Chip Vector Parallelism

Superscalar architectures are unable to efficiently exploit the large number of floating-point units that can potentially be fabricated on a chip, due to the small granularity of their instructions and the correspondingly complex control structure necessary to support it. Vector technology, on the other hand, provides an efficient approach for controlling a large amount of computational resources provided sufficient regularity in the computational structure can be discovered. Vectors exploit these regularities to expedite uniform operations on independent data elements, allowing memory latencies to be masked by overlapping pipelined vector operations with memory fetches. Vector instructions specify a large number of identical operations that may execute in parallel, thus reducing control complexity, reducing power requirements, and efficiently controlling a large amount of computational resources.

However, when such operational parallelism cannot be found, the efficiency of the vector architecture can suffer from the properties of Amdahl's Law where the time taken by the portions of the code that are non-vectorizable can easily dominate the execution time. Thus, vectors are not general-purpose system and cannot effectively process irregularly structured computations that inhibit data-parallel computation. It is therefore critically important to identify the subset of large-scale scientific applications that benefit from the vector paradigm, and quantify performance behavior relative to the coarse-grained parallelism of ultra-scale scalar-based supercomputing platforms. Our previous effort identified application classes both well- and poorly-suited for vectorization, and our proposed work will expand our evaluation suite to examine additional classes of computational methods.

We now give a brief overview of the modern parallel vector architectures that will be examined in our study.

4.4.1 Japanese Earth Simulator

The Earth Simulator (ES) hardly needs an introduction. Since it debuted in the spring of 2002 it has attracted worldwide attention occupying the number one spot on the Top 500 [15] list for a record two and a half years, before moving down to the third position in November 2004. Our research team was the first international group to conduct a performance evaluation study at the Earth Simulator Center [52]; remote ES access in not available. The vector processor of the ES uses a dramatically different architectural approach than conventional cache-based systems. Vectorization exploits regularities in the computational structure of scientific applications to expedite uniform operations on independent data sets. The 1 GHz ES processor contains an 8-way replicated vector pipe capable of issuing a MADD each cycle, for a peak performance of 8.0 Gflop/s per CPU.

The Earth Simulator contains 640 ES nodes (8-way SMPs) connected through a custom single-stage crossbar. This high-bandwidth interconnect topology provides impressive communication characteristics, as all nodes are a single hop from one another. However, building such a network incurs a high cost since the number of cables grows as a square of the node count – in fact, the ES system utilizes approximately 1500 miles of cable. Our study will leverage our close relationship with the ES Center, to continue porting and vectorizing new applications onto this unique platform.

4.4.2 NEC SX-8

Our work will also examine the newly-released NEC SX-8, the world's most powerful vector processor. The SX-8 architecture operates at 2 GHz, and contains four replicated vector pipes for a peak performance of 16 Gflop/s per processor. The SX-8 architecture has several enhancements compared with the ES/SX-6 predecessor, including improved divide performance, hardware square root functionality, and in-memory caching for reducing bank conflict overheads. Both the ES and SX-8 processors contain 72 vector registers each holding 256 doubles, and utilize scalar units operating at the half the peak of their vector counterparts. We plan to conduct experiments on the 72-node SX-8 system at the High Performance Computer Center (HLRS) in Stuttgart, Germany. Preliminary SX-8 experiments have recently been conducted [54].

4.4.3 Cray X1

The Cray X1 is designed to combine traditional vector strengths with the generality and scalability features of modern superscalar cache-based parallel systems. The computational core, called the single-streaming processor (SSP), contains two 32-stage vector pipes running at 800 MHz. Each SSP contains 32 vector registers holding 64 double-precision words (for a vector length of 64), and operates at 3.2 Gflop/s peak for 64-bit data. The SSP also contains a two-way out-of-order superscalar processor running at 400 MHz with two 16KB caches (instruction and data). The multi-streaming processor (MSP) combines four SSPs into one logical computational unit. MSP parallelism is achieved by distributing loop iterations across each of the four SSPs. The compiler must therefore generate both vectorizing and multistreaming instructions to effectively utilize the X1. The scalar unit operates at 1/8th the peak of SSP vector performance, but offers effectively 1/32 MSP performance if a loop can neither be multistreamed nor vectorized. Consequently, a high vector operation ratio is especially critical for effectively utilizing the underlying hardware. Our study will explore the tradeoffs between MSP- and SSP-level parallelism, an open question in the X1 community. Our preliminary work [23, 47, 48, 52, 54, 58] has closely examined several applications on Phoenix, the 512-MSP system at ORNL, and we plan to build on those efforts to further optimize existing codes, while exploring performance of new classes of algorithms.

4.4.4 Cray X1e and BlackWidow

The recently-releases X1e is an evolutionary upgrade of the X1 system. The updated system increases the peak performance by 50% and utilizes double-core MSP processors, for a total of MSPs per node. However, it will be important to understand the effect of these changes, especially in the context memory-intensive code, as the memory bandwidth, cache-size, and SSP performance remain unchanged from the X1. We plan to evaluate the

leadership-class X1e system at ORNL once it becomes available. Additionally, Cray promises to introduce more significant architectural changes to the Cray X2 (BlackWidow). We will leverage our previous investigation of X1 performance to study the efficacy of the X2 architecture on leading scientific codes.

4.5 Understanding Cache Effects for Modern Vector Architectures

An important question our study will address relates to the effectiveness of cache technology for vector architectures. Like traditional vector architectures, the ES vector unit is cache-less; memory latencies are masked by overlapping pipelined vector operations with memory fetches. The X1 architecture, on the other hand, utilizes a 2-way set associative 2MB data Ecache, shared between the four SSPs of an MSP. This configuration allows extremely high bandwidth for X1 computations with temporal data locality. The tradeoffs between these two approaches presents an interesting research question for future vector system development. For example, we have found that for some sets of experiments [58], the X1 performance actually improves when caching is turned off (via software control). Additionally, algorithms which traditionally improve cache-locality — such as data blocking — may decrease the vector length, thus inhibiting vector performance.

We will also study the tradeoffs between the different memory technologies utilized in the ES, SX-8, and X1 system architectures. For example, the main memory chip for the ES uses a specially developed high speed DRAM called FPLRAM (Full Pipelined RAM) operating at 24ns bank cycle time. The SX-8, on the other hand, uses commodity DDR2-SDRAM; thus, we expect higher memory overhead for irregular accesses when compared with the specialized ES memory. However, the SX-8 architecture contains in-memory caching (off-chip) for reducing bank conflict overheads. Our study will measure and analyze the tradeoffs between these varying memory configurations, to help guide application optimization and future system designs.

4.6 Hierarchical System Architectures

The Columbia system, installed at NASA Ames Research Center in the summer of 2004, is currently world's second most powerful supercomputer [15]. This 10,240-processor SGI Altix supercluster is comprised of 20 nodes (or boxes), each containing 512 Intel Itanium2 processors and running the Linux operating system. The Altix system takes the cluster of SMP model to the extreme, utilizing scalable, hardware-supported cache coherent non-uniform memory access system (CC-NUMA) system, which can use 512 (or even more) processors. Each node is supported by the NUMAlink3 interconnect, a high-performance custom network with a fat-tree topology that enables the bisection bandwidth to scale linearly with the number of processors. Our study will first explore the benefits to scientific productivity of utilizing a tightly-integrated shared-memory Altix system.

The Columbia system is unique in that it employs a hierarchical system architectures. The primary communication fabrics used between the 20 Altix systems is an InfiniBand switch, which provides low-latency MPI communication. Additionally, four of the boxes are linked with NUMAlink4 technology to allow global shared-memory constructs with the 2048 processor subsystem. We plan to evaluate this architecture to study the tradeoffs between this hierarchical interconnect infrastructure and the flat network approach employed for massively parallel systems.

5 Target Application Suite

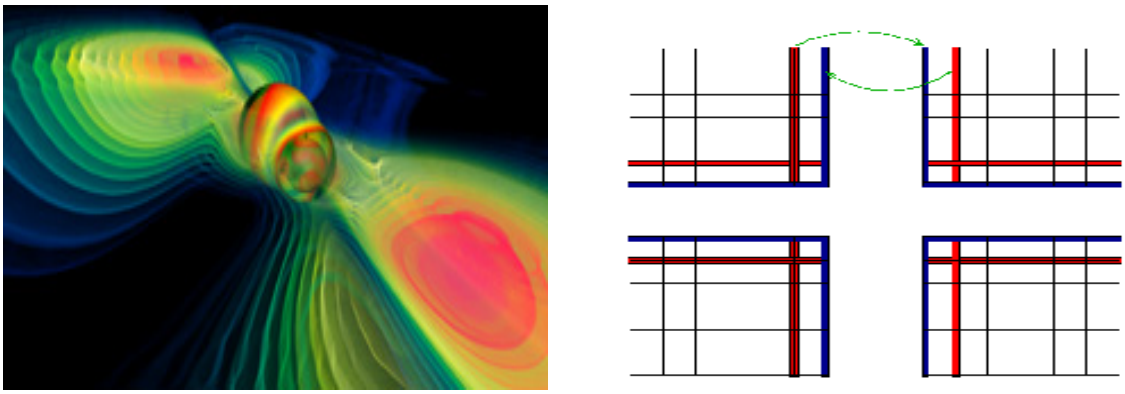
This section presents an overview of the candidate scientific applications evaluated in our study. We have chosen a wide array of algorithms representing some of the most important methods in computational sciences today. Preliminary evaluation work has been performed on several of these applications, allowing us to leverage our code expertise and optimization knowledge to effectively analyze newly-released architectures. We will continue to add new application areas and numerical methods by working closely with domain scientists in their respective field. Table 1 presents an overview of our candidate codes, disciples, numerical methods, and computational structure. Notice that we plan to evaluate a wide variety of modern computational techniques including dense and sparse linear algebra, FFT-based approaches, particle-based algorithms, and adaptive mesh refinement (AMR) calculations.

Name	Lines	Discipline	Problem and Method	Structure
MADCAP	5000	Cosmology	CMB analysis via Newton-Raphson	Dense Matrix
Cactus	84,000	Astrophysics	Einstein's Theory of GR via Finite Differencing	Grid
LBMHD	1,500	Plasma Physics	Magneto-Hydrodynamics via Lattice Boltzmann	Lattice
FVCAM	200,000+	Climate Modeling	Atmospheric Circulation via Finite Volume	Grid
GTC	5,000	Magnetic Fusion	Vlasov-Poisson Equation via Particle in Cell	Particle/Grid
SuperLU	42,000	Linear Algebra	Sparse Solve via LU Decomposition	Sparse Matrix
PMEMD	37,000	Life Sciences	Molecular Dynamics via Particle Mesh Ewald	Particle
PARATEC	50,000	Material Science	Density Functional Theory via FFT	Fourier/Grid
SuperNova	200,000+	Combustion	Rayleigh-Taylor via Adaptive Mesh Refinement	Grid AMR

Table 1: Scientific application suite for evaluation study representing a broad spectrum of numerical methods

5.1 Astrophysics: Cactus

One of the most challenging problems in astrophysics is the numerical solution of Einstein's equations following from the Theory of General Relativity (GR): a set of coupled nonlinear hyperbolic and elliptic equations containing thousands of terms when fully expanded. The Cactus Computational ToolKit [2, 19] is designed to evolve Einstein's equations stably in 3D on supercomputers to simulate astrophysical phenomena with high gravitational fluxes – such as the collision of two black holes and the gravitational waves radiating from that event. Figure 1a presents a visualization of one of the first simulations of the grazing collision of two black holes computed by the Cactus code. The merging black holes are enveloped by their "apparent horizon", which is colorized by its Gaussian curvature. The concentric surfaces that surround the black holes are equipotential surfaces of the gravitational flux of the outgoing gravity wave generated by the collision. The standard MPI driver for Cactus solves the PDE on a local grid section and then updates the values at the ghost zones by exchanging data on the faces of its topological neighbors in the domain decomposition (shown in Figure 1b by the 2D schematic diagram).



(a) (b)

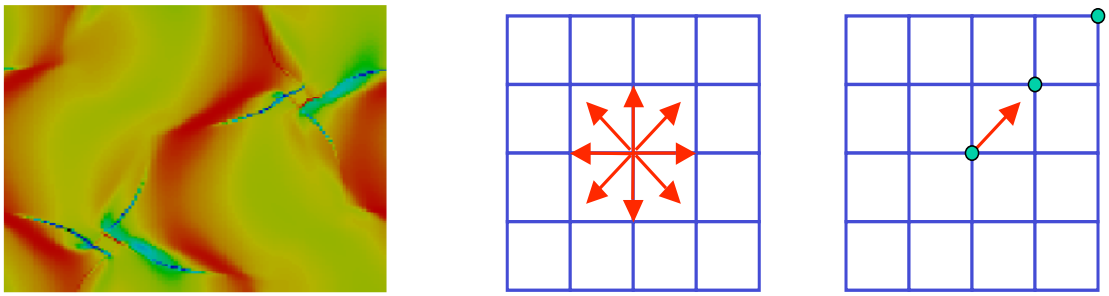
Figure 1: Cactus (a) computed visualization of grazing collision of two black holes. (b) PDEs are solved on the local grid section and then updates ghost zone values on the the faces of its topological neighbors, as show by the 2D schematic diagram.

Our preliminary investigation of Cactus [52] showed that the ES reached an impressive 2.7 Tflop/s (34% of peak) for the largest problem size using 1024 processors. This represents the highest per processor performance (by far) achieved by the production version of Cactus ADM-BSSN on any system to date. Phoenix (X1) ran about 3.8 times slower than the ES, sustaining only 0.7 Gflop/s per processor (6% of peak). This preliminary work highlighted the importance of architectural balance between the vector and scalar units, a question that will be examined in detail as part of our study. We plan to expand our analysis and quantify the effects of scalar–vector

imbalance across our full application suite.

5.2 Plasma Physics: LBMHD

Lattice Boltzmann methods (LBM) have proved a good alternative to conventional numerical approaches for simulating fluid flows and modeling physics in fluids [62]. The basic idea of the LBM is to develop a simplified kinetic model that incorporates the essential physics, and reproduces correct macroscopic averaged properties. Recently, several groups have applied the LBM to the problem of magneto-hydrodynamics (MHD) [30, 43] with promising results. LBMHD [44] simulates the behavior of a two-dimensional conducting fluid evolving from simple initial conditions and decaying to form current sheets. Figure 2a shows the current density decays of two cross-shaped structures after several hundred time steps as computed by LBMHD. The computational structure of LBMHD is shown in Figure 2b. The 2D spatial grid is coupled to an octagonal streaming lattice, as shown in Figure 2b (left) and block distributed over a 2D processor grid.



(a) (b)

Figure 2: LBMHD (a) Visualization of current density decays of two cross-shaped structures after several hundred time steps. (b) Computational structure showing (left) the octagonal streaming lattice coupled with the square spatial grid and (right) example of diagonal streaming vector updating three spatial cells.

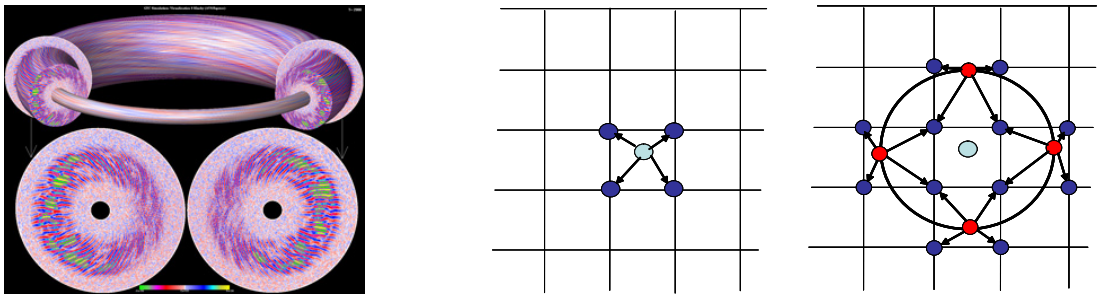
Our preliminary work examined varying schemes in order to optimize the collision routine on for both scalar- and vector-based architectures; details are presented in [52]. Performance measurements on a 8192^2 grid showed impressive ES results, sustaining over 4.6 Gflop/s per processor (58% of peak) — a speedup of 44x, 16x, and 7x compared with Seaborg (Power3), Cheetah (Power4), and on Ram (Altix), respectively — while attaining comparable performance to Phoenix (X1). These results demonstrate that, for this class of applications, the vector platforms sustains a significantly higher fraction of peak, due in part to their superior CPU-memory balance. Our study will examine this issue in depth, especially in the context of newly-released microprocessors, whose CPU-memory balance is substantially higher than previous generations.

Due to the relatively simplicity of LBMHD's computational structure, combined with the significant disparity between scalar and vector performance demonstrated in our work, this code has been chosen as an application benchmark for the DOD HPCS evaluation effort. We plan to continue working closely with the HPCS community in generating full-scale benchmark applications and disseminating our findings to the HPC community at-large.

5.3 Magnetic Fusion: GTC

The Gyrokinetic Toroidal Code (GTC) is a 3D particle-in-cell (PIC) application developed at the Princeton Plasma Physics Laboratory to study turbulent transport in magnetic confinement fusion [41, 42]. . GTC is currently the flagship SciDAC [14] fusion microturbulence code. Turbulence is believed to be the main mechanism by which energy and particles are transported away from the hot plasma core in fusion experiments with magnetic toroidal devices. An in-depth understanding of this process is of utmost importance for the design of future experiments since their performance and operation costs are directly linked to energy losses. Figure 3a shows the 3D visualization of electrostatic potential in global, self-consistent GTC simulation of plasma microturbulence in a magnetic fusion device. The elongated "finger-like" structures are turbulent eddies that act as energy and

particle transport channels. This type of calculation helped to shed light on "anomalous" energy transport that was observed in real experiments. Figure 3b shows the computational structure of the GTC's gyrokinetic PIC approach.



(a) (b)

Figure 3: GTC (a) 3D visualization of electrostatic potential in global, self-consistent simulation of plasma micro-turbulence in a magnetic fusion device. (b) Computational structure showing (left) charge deposition in the classic PIC method and (right) the 4-point averaging of GTC's gyrokinetic PIC approach.

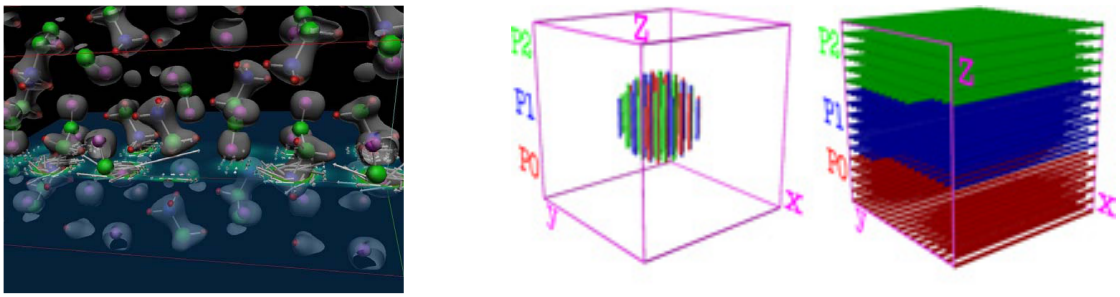
Preliminary experiments were performed using a high resolution domain of 100 particles per cell allowing for significant improvements in the overall statistics of the simulation [52, 53]. The ES achieved 1.6 Gflop/s per processor or 20% of peak on 64 processors — the highest GTC performance on any tested architecture. Phoenix (X1) showed similar performance in absolute terms (1.4 Gflop/s per processor), but a lower fraction of peak at only 11%. Comparing performance with the superscalar architectures, the ES is about 12x, 5.4x, and 5x faster than Seaborg (9% of peak), Cheetah (5%), and Ram (5%), respectively. The vector systems therefore have the potential for significantly higher resolution calculations that would otherwise be prohibitively expensive in terms of time-to-solution on conventional microprocessors. However, this code could potentially be scaled up to very large numbers of processor — we therefore plan to conduct an extensive analysis of the tradeoffs between using a very high level of coarse-grained parallelism (as found in BG/L) and utilizing a relatively small number of processors that leverage fine-grained vector parallelism.

Our proposed work will also examine a new data-decomposition scheme for GTC that, for the first time, enables a breakthrough of the Teraflop barrier. Additionally, we are working to help establish GTC as an HPCS application benchmark [10] due to the importance of this SciDAC fusion application, and its potential to utilize ultra-scale computational resources.

5.4 Material Science: PARATEC

PARATEC (PARAllel Total Energy Code [8]) performs ab-initio quantum-mechanical total energy calculations using pseudopotentials and a plane wave basis set. The pseudopotentials are of the standard norm-conserving variety. Forces can be easily calculated and used to relax the atoms into their equilibrium positions. PARATEC uses an all-band conjugate gradient (CG) approach to solve the Kohn-Sham equations of Density Functional Theory (DFT) and obtain the ground-state electron wavefunctions. Figure 4a shows the induced current and charge density in crystallized glycine, calculated using PARATEC. These simulations are used to better understand nuclear magnetic resonance experiments [66]. In solving the Kohn-Sham equations using a plane wave basis, part of the calculation is carried out in real space and the remainder in Fourier space using specialized parallel 3D FFTs to transform the wavefunctions, a three process example is shown in Figure 4b. Due to PARATEC's global communication requirements, architectures with a poor balance between their bisection bandwidth and computational rate will suffer performance degradation at higher concurrencies.

Preliminary experimental data were gathered for a 432 Silicon-atom bulk system and a standard LDA run of PARATEC with a 25 Ry cut-off using norm-conserving pseudopotentials [47]. Results showed that PARATEC achieves impressive performance on the ES, sustaining 4.7 Gflop/s per processor (58% of peak) on 128 processors. This compares with per processor performance of 0.7 Gflop/s (49%), 1.5 Gflop/s (29%), 3.2 Gflop/s (54%), and 1.9 Gflop/s (15%) on Seaborg, Cheetah, Ram, and Phoenix, respectively. The ES outperformed all other platforms



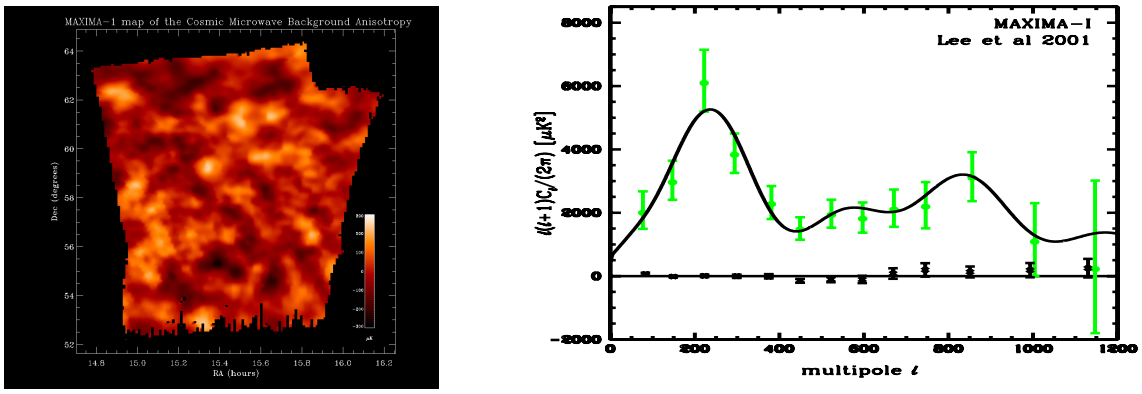
(a) (b)

Figure 4: PARATEC (a) Visualization of induced current (white arrows) and charge density (colored plane and gray surface) in crystallized glycine, described in [66]. (b) Computational structure showing a three-processor example of PARATEC's parallel data layout for the wavefunctions of each electron in (left) Fourier space and (right) real space.

primarily due to a superior architectural balance of bisection bandwidth relative to computation rate — a critical component for achieving high scalability during the global grid transformation of the wavefunctions. Our study will expand our evaluation of PARATEC across a broad spectrum of architectural platforms. This application is an extremely useful tool in evaluating the balance between computational resources and global interprocessor communication facilities, and will be a critical component of our study in evaluating the tradeoffs of evolving interconnect designs.

5.5 Cosmology: MADCAP

The Cosmic Microwave Background (CMB) is a snapshot of the Universe some 400,000 years after the Big Bang. The pattern of anisotropies in the CMB carries a wealth of information about the fundamental parameters of cosmology. Realizing the extraordinary scientific potential of the CMB requires making precise measurements of the microwave sky temperature over a significant fraction of the sky at very high resolution. Such measurements are made by scanning the sky for as long as possible with a cryogenically cooled telescope and as many microwave detectors as possible. The reduction of the resulting datasets — first to a pixelized sky map, and then to an angular power spectrum (see Figure 5) — is a serious computational challenge, and one which is only getting worse with increasing dataset sizes, as we try to make ever more precise measurements. It is therefore critical to choose the optimal algorithmic approach and supercomputing platform; one approach is the Microwave Anisotropy Dataset Computational Analysis Package (MADCAP) [22], which has been widely used on a variety of platforms [25].



(a) (b)

Figure 5: MADCAP (a)Input map data and (b) associated angular power spectrum calculation of the CMB sky measured by the MAXIMA experiment [38]

The full MADCAP spectral estimator includes a large number of special-case features, from preliminary data checking to marginalization over foreground templates, that dramatically increase the size and complexity of the code without altering its basic operational structure. For simplicity, we are therefore developing a stripped-down version, called MADbench [23], a lightweight version of the MADCAP expressly designed for benchmarking, that retains the operational complexity and integrated system requirements of the full application. Preliminary MADbench results are presented in Section 6.2. We plan to publicly distribute MADbench, and use our synthetic benchmark generation methodology as a model for creating full-scale portable codes for community-wide evaluation efforts. Additionally, we are in the process of distributing MADbench to the HPCS community [10], as it combines the potential for ultra-scale parallelism with the challenges of heavy I/O requirements. Several other DOE-supported groups including the FastOS research community [11], have shown interest in in MADbench, and we plan to work closely together to support their research effort. Finally, utilizing MADbench on the latest generation of HEC architectures, will allow us to understand the system balances of various platforms, especially in the context of I/O — a critical issue for future data-intensive analyses.

5.6 Climate Modeling: FVCAM

The Community Atmosphere Model (CAM) is the atmospheric component of the flagship Community Climate System Model (CCSM3.0). Developed at the National Center for Atmospheric Research (NCAR), the CCSM3.0 is extensively used to study climate change. The CAM application is an atmospheric general circulation model (AGCM) and can be run either coupled within CCSM3.0 or in a stand-alone mode driven by prescribed ocean temperatures and sea ice coverages [28]. AGCMs are key tools for weather prediction and climate research. They also require large computing resources: even the largest current supercomputers cannot keep pace with the desired increases in the resolution of these models.

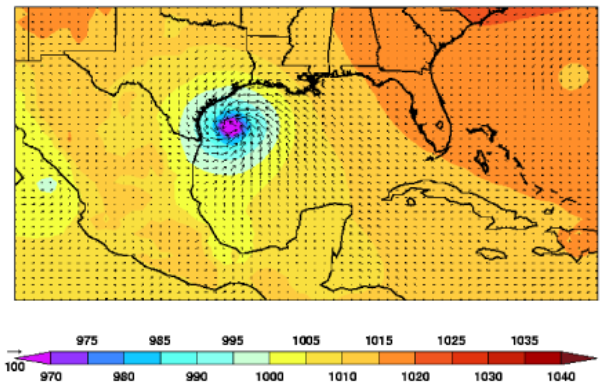


Figure 6: FVCAM simulation using the D mesh of a Class IV hurricane about to reach the coast of Texas.

Our group is the first to present vector results of the Community Atmosphere Model using the finite-volume solver in the dynamics phase of the calculation. Preliminary results on FVCAM [54] compares Seaborg and the ES for a high resolution $0.5°\text{x}0.625°$ horizontal mesh configuration (sometimes labeled as the D grid) — corresponding to 576 longitudinal grid points and 361 latitudinal grid points. Results show that at high concurrencies, Seaborg and the ES sustain similarly low percentages of peak (around 4%). Our proposed work will continue the investigation of FVCAM across a broad spectrum of computing platform using very-large grid configurations and thousands of processors, with the goal of isolating the performance-limiting architectural features. Due to the limited number of coarse-grained domain decompositions, FVCAM represents an example of a code that may not benefit from ultra-parallel systems, but must instead utilize fine-grained parallelism such as vectorization. Our study will focus on understanding these tradeoffs for the latest HEC platforms.

5.7 Combustion: AMR

A Type Ia supernova explosion likely begins as a nuclear runaway near the center of a carbon-oxygen white dwarf. The outward propagating flame is unstable to the Landau-Darrieus, Rayleigh-Taylor, and Kelvin-Helmholtz instabilities, which serve to accelerate the flame to a large fraction of the speed of sound. At present, the exact mechanism for the subsequent explosion of the star is unknown and investigators have turned to numerical simulation. The SuperNova [20] code, developed by researchers in the Center for Computational Sciences and Engineering at LBNL, is one such code that models the unstable flow processes in reacting, degenerate nuclear matter that occurs in white dwarf stars. A visualization of a three-dimensional unstable carbon flame, as calculated by SuperNova, is shown in Figure 7. SuperNova utlizes Adaptive Mesh Refinement (AMR): a technique for automatically refining (or de-refining) certain regions of the physical domain concentrating computational resources where interesting physics is occurring.

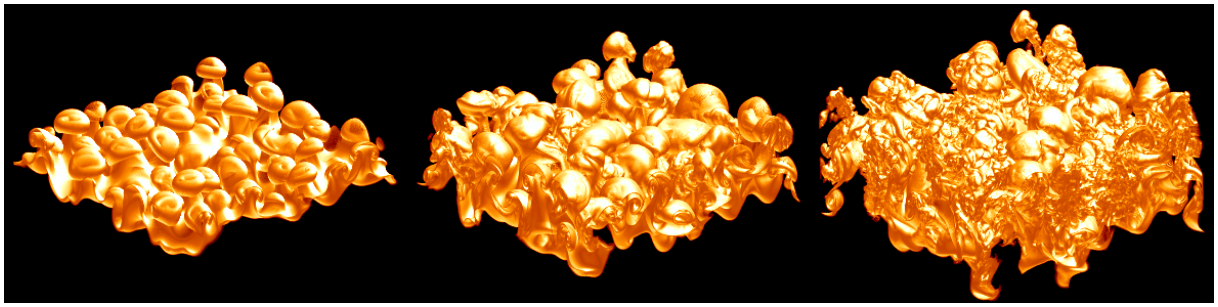


Figure 7: Three-dimensional Rayleigh-Taylor unstable carbon flame. Three views of the carbon mass fraction are shown, from early to late stages.

Vectorization of AMR-based computations presents a tremendous challenge due to the non-vectorizable overheads associated with management of the hierarchical data structures and dynamic load balancing overheads. However, the underlying numerical computations contain data-parallel operations, which are expected to perform well on vector systems. Measuring and analyzing these tradeoffs is a critical step in understanding the potential of effectively using vector architectures for future AMR computations. It is commonly believed that adaptive calculations will become a key component of future high-fidelity multi-scale simulations, across of broad spectrum of application domains. However, to date, no AMR performance results on modern parallel vector architectures are available to the scientific community. Our proposed work will conduct this important evaluation.

5.8 Linear Algebra: SuperLU

An emerging method for scientific computation is the use of sparse matrix methods. We plan to explore this important class of algorithms. Our current candidate is SuperLU. SuperLU [40] is a general purpose library for the direct solution of large, sparse, nonsymmetric systems of linear equations on high performance machines. The library routines performs an LU decomposition with partial pivoting and triangular system solves through forward and back substitution. The LU factorization routines can handle non-square matrices but the triangular solves are performed only for square matrices. The matrix columns may be preordered (before factorization) either through library or user supplied routines. This preordering for sparsity is completely separate from the factorization. Working precision iterative refinement subroutines are provided for improved backward stability. Routines are also provided to equilibrate the system, estimate the condition number, calculate the relative backward error, and estimate error bounds for the refined solutions. The irregular data access patterns of this code presents a performance challenge for superscalar systems, while the complexity of the control flow is expected to inhibit vector performance. This class of codes is at odds with traditional cache-based architectures, and will allow us to evaluate the tradeoffs between various classes of memory hierarchies.

5.9 Life Sciences: PMEMD

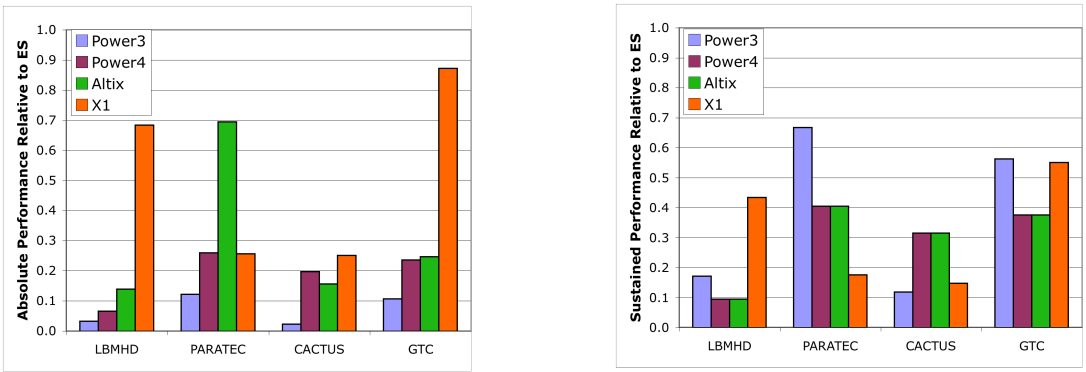
Finally, we are interested in investigating algorithms in the field of molecular dynamics, due to their increasing importance and computational irregularity. One candidate code is PMEMD. PMEMD [12] an application that performs molecular dynamics (MD) simulations and minimizations using Particle mesh Ewald molecular dynamics. The force evaluation is performed in an efficiently parallel manner using state of the art numerical and communication methodologies. PMEMD uses a highly asynchronous approach to communication for the purposes of achieving a high degree of parallelism. Variants on the parallel particle mesh calculations in PMEMD are used in several other MD codes. The dynamic nature of this code is at odds inhibits data-parallelism and will present a significant challenge for vector architectures. This application may therefore benefit more from ultra-parallel systems than architectures which depend on fine-grained chip-level parallelism. Our study will investigate these competing tradeoffs.

6 Preliminary Results

Our study will generate numerous types of performance analyses using the broad spectrum of examined HEC platforms and application domains. Here we show three preliminary examples: direct architectural performance evaluation, comparison of relative system balance behavior, and investigation of high-end application communication requirements.

6.1 Evaluation of Architecture Performance

This section presents a brief performance overview from our preliminary evaluation work. Results are shown using four diverse scientific applications (LBMHD, Cactus, GTC, and PARATEC) on the parallel vector architectures of the ES and X1, and three leading superscalar platforms, the Power3, Power4, and Altix. Since most modern scientific codes are designed for (super)scalar microprocessors, it was necessary to port these applications onto the vector platforms; code transformations were applied in an attempt to maximize the vector operation ratio and average vector length. The current proposal aims to extend these experiments to a wider class of applications and numerical methods.



(a) (b)

Figure 8: Preliminary performance results of LBMHD, CACTUS, PARATEC, and GTC on the ES, Power3 (Seaborg), Power4 (Cheetah), Altix (Ram) and X1 (Phoenix) showing (a) absolute performance as a ratio of the ES and (b) sustained performance (percentage of peak) as a ratio of the ES.

Figure 8 shows (a) absolute and (b) sustained performance relative to the ES (using the largest comparable concurrency for each architecture). Results show that the ES vector system achieved excellent performance on our application suite – the highest of any architecture tested to date – demonstrating the tremendous potential of modern parallel vector systems. The ES consistently sustained a significantly higher fraction of peak than the X1, due in

part to superior scalar processor performance, memory bandwidth, and network bisection bandwidth relative to the peak vector flop rate. A number of performance bottlenecks exposed on the vector machines relate to the extreme sensitivity of these systems to small amounts of unvectorized code. This sheds light on a different dimension of architectural balance than simple bandwidth and latency comparisons. Out study will continue building on these efforts on the latest HEC platforms.

6.2 Evaluation of System Balance

Our next set of preliminary results present a detailed analysis of the MADbench cosmology performance characterization that help identify the relative system balance in terms of computation, interprocessor data remapping, MPI communication, and I/O overhead.

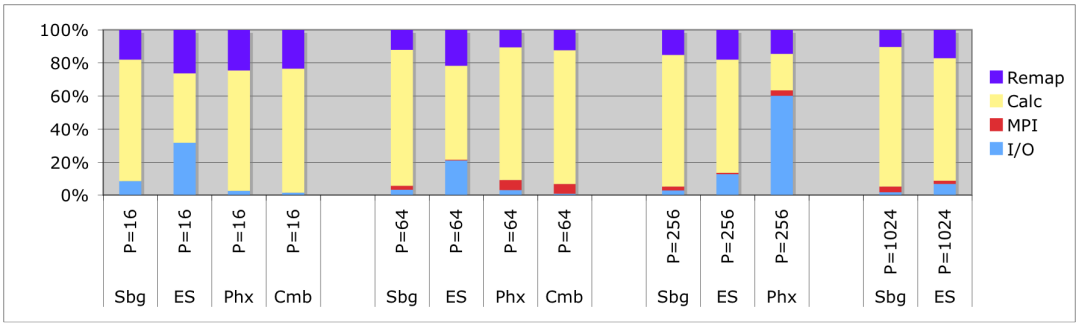


Figure 9: Relative performance breakdown (in percentage of peak) of MADbench on the Power3 (Sbg), ES, X1 (Phx), Columbia (Cmb), highlighting the relative balance of each platform

Figure 9 illustrates the percentage of theoretical peak performance obtained on each architecture, at each concurrency, for the full MADbench code on the ES, X1, Power3, and Columbia. Each entry actually consists of four overlayed bars showing the percentage of peak achieved by considering (i) all the run time, (ii) all but the remapping time, (iii) all but the remapping and I/O time, and (iv) all but the remapping, I/O and MPI time. Another way to interpret this figure is that the amount of purple visible reflects the relative cost of remapping[1], red the relative cost of I/O, and yellow the relative cost of MPI. In this light it is clear that I/O is a minor issue for Seaborg, but a very significant issue for the ES at all concurrencies, for Phoenix 64-way or more, and for Columbia 256-way; interestingly, the cost is broadly flat for Seaborg, significantly decreases with increasing concurrency for the ES (reflecting its slow but perfectly scalable filesystem), but dramatically increases with concurrency for Phoenix and Columbia. It is also apparent that MPI is a relatively small cost for all but Phoenix, but that it is significant and increases with concurrency there.

A more general conclusion of this work is that greater clarity in the application context and overall specificity of performance timings greatly benefit understanding of how the distinct parts of large-scale parallel application interact with the distinct subsystems of HEC platforms. Often in real world cases it is not sufficient to report only the overall timing for a full–blown parallel application in order to understand overall performance.

Since most scientific applications have distinct code phases, which utilize many distinct subsystems of a parallel computer, the achieved performance will not approximate that seen in simple computational kernels which model only the compute phase and often vastly over estimate sustained performance. Such in-depth analysis is critical to resolving the discrepancy between theoretical and sustained parallel performance. Our proposed work will extend this type of analysis to a wide variety of computational methods and architectural platforms.

[1]Columbia 256-way excepted due to bug in the Altix ScaLAPACK remapping function, SGI engineers are addressing this problem.

6.3 Communication Analysis

In this section we describe our proposed communication analysis study, highlighting its important and preliminary methodology. As a preliminary step in our proposed work, we explore the communication topology of six applications described in Section 5. Our proposed work will examine additional communication characteristics, such as bandwidth and latency requirements, which are of key interest to network designers and system software developers.

In order to profile the communication characteristics of scientific applications for our study, we employ the Integrated Performance Monitoring (IPM) [60] tool, recently developed at LBNL. IPM is an application profiling layer that allows us to non-invasively gather the communication characteristics of these codes as they are run in a production environment. IPM maintains low overhead by using a unique hashing approach which allows a fixed memory footprint and minimal CPU usage, allowing it to trivially scale to thousands of tasks.

Here we present the topological connectivity for each application by representing the volume and pattern of message exchanges between all tasks. By recording statistics on these message exchanges we can form an undirected graph which describes the topological connectivity required by the application. This graph is undirected because we assume that switch links are bi-directional. As a result, the topologies shown are always symmetric about the diagonal. From this graph we can calculate certain reduced quantities which describe the communication pattern at a coarse level. Such reduced metrics are important in being able to make direct comparisons between applications. Additionally, we examine the average *topological degree of communication* (TDC) of each code, a key metric for evaluating the potential benefit from low-connectivity interconnect technology.

6.3.1 Preliminary Study of Communication Topology

An overview of the volume and topology of interprocessor communication for GTC, Cactus, LBMHD, SuperLU, PMEMD, and PARATEC are shown in Figure 10.

Figure 10(a), shows that GTC has a regular communication structure. This particle-in-cell calculation uses a one-dimensional domain decomposition across the toroidal computational grid, causing each processor to exchange data with its two neighbors as particles cross the left and right boundaries. Additionally, there is a particle decomposition within each toroidal partition, resulting in an average TDC of 4. This small TDC requirements clearly indicate that most links on a fully-connected network would not be utilized for the GTC simulation.

Figure 10(b), shows that that the ghost-zone exchanges of Cactus result in communications with 'neighboring' nodes, represented by diagonal bands. In fact, each node communicates with at most 6 neighbors due to the regular computational structure of this 3D stencil code. On average, the TDC is 5, because some nodes are on the boundary and therefore have fewer communication partners. Note, that the low TDC indicates an opportunity to effectively utilize a low-connectively network.

The connectivity of LBMHD is shown in Figure 10(c). Structurally, we see that the communication, unlike Cactus, is scattered (not occurring on the diagonal). This is due to the interpolation between the diagonal streaming lattice and underlying structure grid. Note that although the 3D LBMHD streams the data in 27 directions, the code is optimized to reduce the number of communicating neighbors to 12. Once again this code has the potential to map well onto a low-connectively network.

Figure 10(d) shows the connectivity and TDC for our SuperLU runs. The complex communication structure of this computation results in many point-to-point message transmissions, where the topological connectively is a function of concurrency, scaling proportionally to the $\sqrt{P}$. Thus, at first glance, SuperLU seems to be a poor candidate for low-interconnect networks.

Figure 10(e) shows the complex communication structure of the PMEMD particle mesh ewald calculation. Here the average TDC is equal to P and the degree of connectivity is a function of concurrency. For the spatial decomposition used in this algorithm, each task's data transfer with another task drops off as their spatial regions become more distant. The rate of this drop off depends strongly on the molecule(s) in the simulation. As a result, there is a significant disparity between the maximum and average TDP. This disparity would inhibit high performance on traditional low-connectivity networks; however, several research teams are exploring dynamically reconfigurable network solutions [26, 32, 37], which may be applicable to this class of applications.

Finally, Figure 10(f) shows the communication requirements of PARATEC. This communication-intensive code relies on global data transposes during its 3D FFT calculations, resulting in large, global message traffic [24].

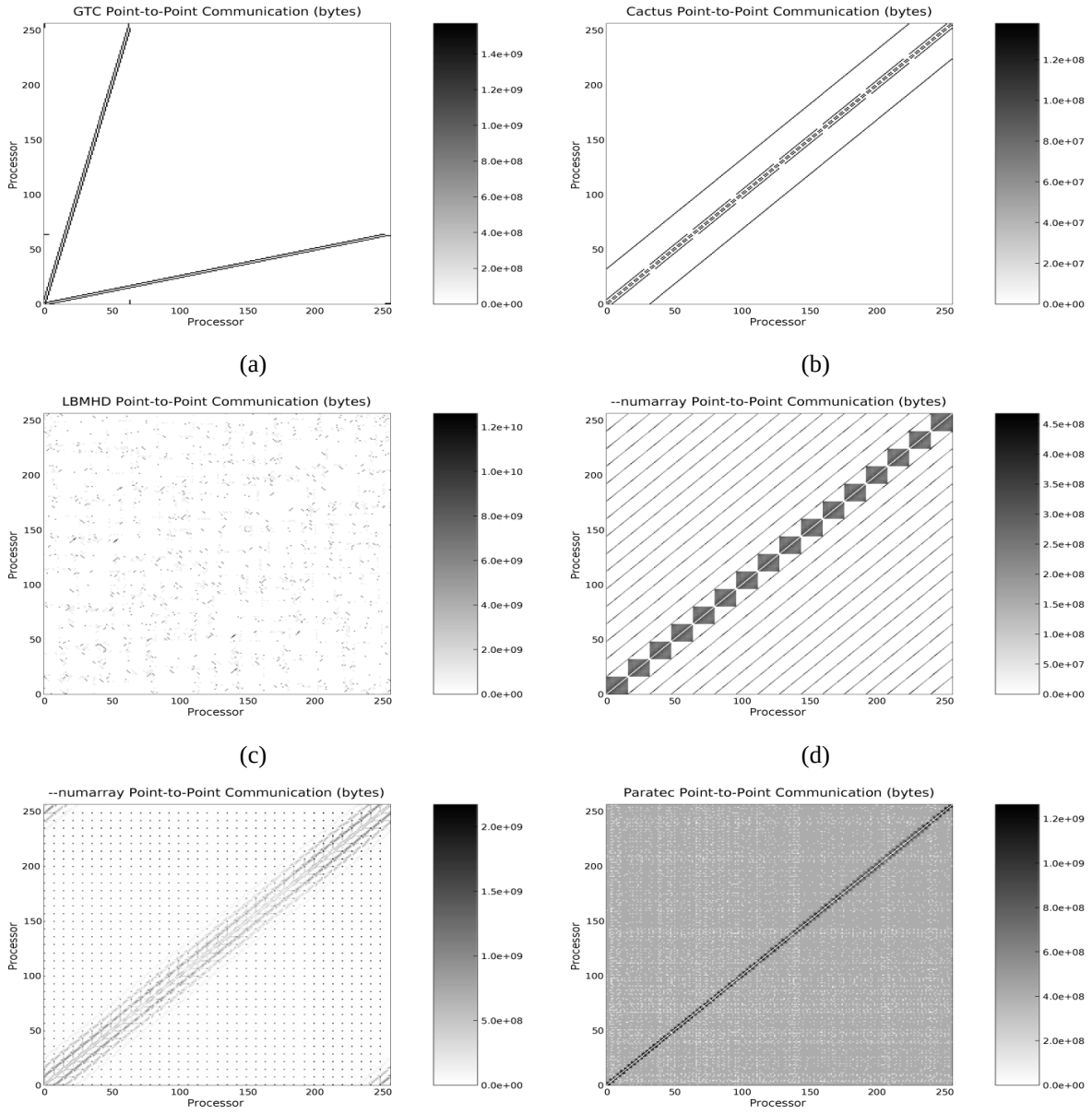


Figure 10: Volume and topology of interprocessor communication using 256 processors for (a) GTC and (b) Cactus (c) LBMHD (d) SuperLU (e) PMEMD and (f) PARATEC.

Thus, PARATEC represents the class of codes that requires use of the bisection bandwidth that a fully-connected network configuration can provide — as a result low-connectivity networks are inappropriate for this class of applications.

6.3.2 Communication Analysis Summary

Most high-end computing platforms are currently connected via fully-interconnected networks whose cost grows superlinearly relative to system size — making these network solutions impractical for next-generation peta-flop platforms employing tens (or hundreds) of thousands of processors. In order to understand the feasibility of employing lower-connectively networks, a comprehensive study exploring the potential mapping between high-end application requirements and low-connectivity interconnect designs (n-dimensional tori, hypercubes, etc) is required. Our preliminary investigation analyzed six full-scale scientific codes, with widely varying computational methods, and has shown a promising direction in quantifying the communication topological requirements of leading applications. We plan to leverage our application expertise to build on this approach, and develop a robust methodology for understanding application requirements — thus contributing to the effective design and utilization of next-generation ultra-parallel systems.

7 Relevant Research Experience

The Dr. Leonid Oliker has considerable experience evaluating a wide range of architectural platforms and programming paradigms, and has received Best Paper Awards at SC99 [46] and SC2000 [57] for work in this area. Recently he has been involved in extensive analysis of existing vector platforms[23, 47, 48, 52, 53, 54, 58] such as the NEC SX-6, and the Cray X1,as well as leading the first international team to evaluate the Earth Simulator System. Additionally, he has a strong background in the evaluation of experimental and emerging architectural designs. Examples include performance analysis of the Tera MTA [46, 49, 51], the Berkeley VIRAM vector processor [34], based processor-in-memory technology, and the Stanford Imagine processor [27, 36], which uses a variation of vectors called streams. For each given technology, new algorithmic approaches were developed to fully leverage the underlying architectural features on a set of key computational kernels. Dr. Oliker is also participating in research activities investigating the performance-limiting microarchitectural bottlenecks of leading superscalar processor designs [35, 39]. Finally, he has conducted a number studies examining the complex interplay between high-level algorithms, programming paradigms, and architectural platforms [21, 50, 56, 57].

Professor Katherine Yelick's has an extensive research background in high performance computing, parallel programming languages, compiler analyses for explicitly parallel code, and optimization techniques for communication and memory system architectures. Her parallel language and compiler projects include Split-C and UPC, which are parallel extensions of C, and Titanium, a high performance scientific computing language based on Java. She led the compiler effort for the Berkeley IRAM project, a single chip system that combines vector processing computing in a Processor-in-Memory chip, and the Sparsity code generation system for automatic tuning of sparse matrix kernels. She currently leads the Future Technologies Group and UPC team at LBNL and co-leads the Titanium and BeBOP (Berkeley Benchmarking and Optimization) teams at the University of California, Berkeley. She is also the Director of the Berkeley Institute for Performance Studies, a joint LBNL and UCB effort.

8 Milestones

Our proposed study will examine a broad range of important DOE applications utilizing state-of-the-art numerical methods, and expanding our previous application suite to include some of the more challenging methods to parallelize. We plan to continue our close collaboration with application scientist in selecting the suite of evaluated codes, perform detailed analyses of underlying application requirements, examine newly-emerging architectural platforms, and distribute benchmark codes that preserve the full computational complexity of the underlying scientific application. Our effort will focus on the use of ultra-scale, low power systems, and will evaluate the need for high degree networks as well as tightly integrated networks. We will also continue exploring some of the questions related to single node performance, including the use of vector processors vs. commodity nodes, on the

new applications in our study. These results will help network designers and system procurement teams prepare for next-generation ultra-parallel computing platforms. Our work will be disseminated to the HPC community at large, via publications in leading workshops, conferences, and academic journals. Our research project has the following major milestones:

Year1

- Conduct extensive performance evaluation of the largest clusters built form commodity nodes, such as Itanium/Quadrics and Opteron/Infiniband clusters, and identify the application classes best suited for each given class of architecture.
- Study the behavior of applications on hierarchical system interconnects in constellation (supercluster) architectures and compare them with flat, ultra-scale system architecture.
- Introduce life science application into evaluation suite.

Year2

- Introduce complex, irregular application classes requiring either AMR or sparse matrix methods into the applications suite.
- Perform extensive analysis of high-end application communication requirements, and determine what degree of interconnect connectivity is suitable for each application class.
- Conduct in-depth study of application suite on low-power, ultra-parallel computing platforms and compare performance with the finer-grained on-chip parallelism.

Year3

- For each class of applications, identify the required system balance needed to match the needs of the application.
- Utilize topological communication requirement data to determine when and how to map applications onto low-degree connectivity networks.
- Explore next-generation architectures, as they become available, and identify their suitability for the classes of applications in our suite.

References

[1] Advanced Scientific Computing Research FY2006 Budget. `http://www.sc.doe.gov/orm/Budget_Finance/FY_06_Budget/ASCR.pdf`.

[2] Cactus Code Server. `http://www.cactuscode.org`.

[3] DOD High Performance Computing Modernization Program. `http://www.hpcmo.hpc.mil/`.

[4] DOD High Productivity Computing Systems (HPCS) . `http://www.darpa.mil/ipto/programs/hpcs/`.

[5] High End Computing University Research Activity. `http://www.itrd.gov/subcommittee/hec/hecrtf-outreach/hec-ura/`.

[6] NAS Parallel Benchmarks. `http://www.nas.nasa.gov/Software/NPB`.

[7] ORNL Evaluation of Early Systems. `http://www.csm.ornl.gov/evaluation/`.

[8] PARAllel Total Energy Code. http://www.nersc.gov/projects/paratec.

[9] Performance and Architecture Laboratory, Los Alamos National Laboratory. http://www.c3.lanl.gov/par_arch/.

[10] Personal communication Dave Koester, HPCS Productivity Team, 2004.

[11] Personal communication Patricia Teller, FASTOS PI, 2004.

[12] PMEMD: Particle mesh Ewald molecular dynamics. http://amber.scripps.edu/pmemd-get.html.

[13] SciDAC Performance Evaluation Center. http://perc.nersc.gov/.

[14] SciDAC: Scientific Discovery through Advanced Computing. http://www.scidac.org/.

[15] Top500 Supercomputer Sites. http://www.top500.org.

[16] Science case for large-scale simulation. In D. Keyes, editor, *DOE Office of Science Workshop*, June 2003.

[17] Workshop on the roadmap for the revitalization of high-end computing. In D.A. Reed, editor, *Computing Research Association*, June 2003.

[18] P. A. Agarwal et al. Cray X1 evaluation status report. In *Proc. of the 46th Cray Users Group Conference*, May 17-21, 2004.

[19] M. Alcubierre, G. Allen, B. Brgmann, E. Seidel, and W.-M. Suen. Towards an understanding of the stability properties of the 3+1 evolution equations in general relativity. *Phys. Rev. D*, (gr-qc/9908079), 2000.

[20] J. B. Bell, M. S. Day, C. A. Rendleman, S. E. Woosley, and M. A. Zingale. Direct numerical simulations of type Ia Supernovae flames II: The Rayleigh-Taylor instability. *Astrophysical Journal*, 608, 2004.

[21] R. Biswas, S. Das, D. Harvey, and L. Oliker. Parallel dynamic load balancing strategies for adaptive irregular applications. *Applied Mathermatical Modeling Journal*, 25, 2000.

[22] J. Borrill. MADCAP: The Microwave Anisotropy Dataset Computational Analysis Package. In *5th European SGI/Cray MPP Workshop*, Bologna, Italy, 1999.

[23] J. Borrill, J. Carter, L. Oliker, D. Skinner, and R. Biswas. Integrated performance monitoring of a cosmology application on leading hec platforms. In *ICPP: International Conference on Parallel Processing*, Oslo, Norway, 2005, to appear.

[24] A. Canning, L.W. Wang, A. Williamson, and A. Zunger. Parallel empirical pseudopotential electronic structure calculations for million atom systems. *J. Comput. Phys.*, 160:29, 2000.

[25] J. Carter, J. Borrill, and L. Oliker. Performance characteristics of a cosmology package on leading hpc architectures. In *HiPC:International Conference on High Performance Computing*, Bangalore, India, 2004.

[26] Roger D. Chamberlain, Ch'ng Shi Baw, and Mark A. Franklin. Gemini: An optical interconnection network for parallel processing. *IEEE Trans. on Parallel and Distributed Systems*, 13, October 2002.

[27] S. Chatterji, J. Duell, M. Narayanan, and L. Oliker. Performance evaluation of two emerging media processors: Viram and imagine. In *Workshop on Parallel and Distributed Image Processing, Video Processing, and Multimedia*, Nice, France, 2003.

[28] W. D. Collins et al. The Formulation and Atmospheric Simulation of the Community Atmosphere Model: CAM3. *Journal of Climate*, to appear, 2005.

[29] K. Davis, A. Hoisie, G. Johnson, D. Kerbyson, M. Lang, S. Pakin, and F.Petrini. A performance and scalability analysis of the BlueGene/L architecture. In *Proc. SC2004: High performance computing, networking, and storage conference*, Pittsburgh, PA, Nov6-12, 2004.

[30] P.J. Dellar. Lattice kinetic schemes for magnetohydrodynamics. *J. Comput. Phys.*, 79, 2002.

[31] T. H. Dunigan Jr., M. R. Fahey, J. B. White III, and P. H. Worley. Early evaluation of the Cray X1. In *Proc. SC2003: High performance computing, networking, and storage conference*, Phoenix, AZ, Nov 15-21, 2003.

[32] Hans Eberle and Nils Gura. Separated high-bandwidth and low-latency communication in the cluster interconnect clint. In *Proceedings of the IEEE Conference on Supercomputing*, 2002.

[33] M. Fahey and J. Candy. GYRO: A 5-D gyrokinetic-Maxwell solver. In *Proc. SC2004: High performance computing, networking, and storage conference*, Pittsburgh, PA, Nov6-12, 2004.

[34] B. Gaeke, P. Husbands, L. Oliker, X. Li, K. Yelick, and R. Biswas. Memory-intensive benchmarks: Iram vs. cache-based machines. In *IPDPS: International Parallel and Distributed Processing Symposium*, Fort Lauderdale, Fl, 2002.

[35] G. Griem, L. Oliker, J. Shalf, and K. Yelick. Identifying performance bottlenecks on modern microarchitectures using an adaptable probe. In *Proc. 3rd International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS)*, Santa Fe, New Mexico, Apr. 26-30, 2004.

[36] Gorden Griem and Leonid Oliker. Transitive closure on the imagine stream processor. In *Workshop on Media and Stream Processors*, San Diego, CA, 2004.

[37] Vipul Gupta and Eugen Schenfeld. Performance analysis of a synchronous, circuit-switched interconnection cached network. In *ICS '94: Proceedings of the 8th international conference on Supercomputing*, pages 246–255, New York, NY, USA, 1994. ACM Press.

[38] S. Hanany et al. Maxima-1: A measurement of the cosmic microwave background anisotropy on angular scales of $10'$–5°. *The Astrophysical Journal*, 545:L5–L9, 2000.

[39] S. Kamil, P. Husbands, L. Oliker, John Shalf, and K. Yelick. Impact of modern memory subsystems on cache optimizations for stencil computations. In *3rd Annual ACM SIGPLAN Workshop on Memory Systems Performance*, Chicago,IL, 2005, to appear.

[40] Xiaoye S. Li and James W. Demmel. Superlu–dist: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Trans. Mathematical Software*, 29(2):110–140, June 2003.

[41] Z. Lin, S. Ethier, T.S. Hahm, and W.M. Tang. Size scaling of turbulent transport in magnetically confined plasmas. *Phys. Rev. Lett.*, 88, 2002.

[42] Z. Lin, T. S. Hahm, W. W. Lee, W. M. Tang, and R. B. White. Turbulent transport reduction by zonal flows: Massively parallel simulations. *Science*, Sep 1998.

[43] A. Macnab, G. Vahala, P. Pavlo, , L. Vahala, and M. Soe. Lattice boltzmann model for dissipative incompressible MHD. In *Proc. 28th EPS Conference on Controlled Fusion and Plasma Physics*, volume 25A, Funchal, Portugal, June 18-22, 2001.

[44] A. Macnab, G. Vahala, L. Vahala, and P. Pavlo. Lattice boltzmann model for dissipative MHD. In *Proc. 29th EPS Conference on Controlled Fusion and Plasma Physics*, volume 26B, Montreux, Switzerland, June 17-21, 2002.

[45] K. Nakajima. Three-level hybrid vs. flat mpi on the earth simulator: Parallel iterative solvers for finite-element method. In *Proc. 6th IMACS Symposium Iterative Methods in Scientific Computing*, volume 6, Denver, Colorado, March 27-30, 2003.

[46] L. Oliker and R. Biswas. Parallelization of a dynamic unstructured application using three leading paradigms. In *Proc. SC99: High performance computing, networking, and storage conference*.

[47] L. Oliker, R. Biswas, J. Borrill, A. Canning, J. Carter, J. Djomehri, H. Shan, and D. Skinner. A performance evaluation of the Cray X1 for scientific applications. In *VECPAR: 6th International Meeting on High Performance Computing for Computational Science*, Valencia, Spain, June 28-30, 2004.

[48] L. Oliker, R. Biswas, Rob Van der Wijngaart, David Bailey, and Allan Snavely. Performance evaluation and modeling of ultra-scale systems. *Frontiers of Parallel Processing for Scientific Computing*, 2005, to appear.

[49] L. Oliker, R. Biswas, P. Husbands, and X. Li. Effects of ordering strategies and programming paradigms on sparse matrix computations. *Siam Review*, 44:3, 2002.

[50] L. Oliker, R. Biswas, X. Li, and G. Heber. Parallel conjugate gradient: Effects of ordering strategies, programming paradigms, and architectural platforms. In *IPDPS:Interational Conference on Parallel and Distributed Computing Systems*, Cancun, Mexico, 2000.

[51] L. Oliker, R. Biswas, and H. Shan. Parallel computing strategies for irregular algorithms. *Annual Review of Scalable Computing*, 5, 2003.

[52] L. Oliker, A. Canning, J. Carter, and J. Shalf. Scientific computations on modern parallel vector systems. In *Proc. SC2004: High performance computing, networking, and storage conference*, Pittsburgh, PA, Nov6-12, 2004.

[53] L. Oliker, A. Canning, J. Carter, J. Shalf, D. Skinner, S. Ethier, R. Biswas, M.J. Djomehri, , and R.F. Van der Wijngaart. Performance evaluation of the sx-6 vector architecture for scientific computations. *Concurrency and Computation; Practice and Experience*, 17:1:69–93, 2005.

[54] L. Oliker, J. Carter, M. Wehner, A. Canning, S. Ethier, B. Govindasamy, A. Mirin, and D. Parks. Leading computational methods on
scalar and vector hec platforms. In *SC2005: High performance computing, networking, and storage conference*, Seattle, WA, 2005, submitted. Available at: `http://crd.lbl.gov/~oliker/drafts/SC05_ES_submit.pdf`.

[55] T. Pohl, F. Deserno, N. Thurey, U. Rude, P. Lammers, G. Wellein, and T. Zeiser. Performance evaluation of parallel large-scale lattice bolzmann applications on three supercomputing architectures. In *Proc. SC2004: High performance computing, networking, and storage conference*, Pittsburgh, PA, Nov6-12, 2004.

[56] H. Shan, J. Singh, and R. Biswas. Message passing and shared address space parallelism on an smp cluster. *Parallel Computing Journal*, 29:2, 2003.

[57] H. Shan, J. Singh, L. Oliker, and R. Biswas. A comparison of three programming models for adaptive applications on the origin2000. In *Proc. SC2000: High performance computing, networking, and storage conference*.

[58] H. Shan, E. Strohmaier, and L. Oliker. Optimizing performance of superscalar codes for a single cray x1 msp. In *46th Cray User Group Conference*, Knoxville, TN, 2004.

[59] H.D. Simon, W.T. Kramer, W. Saphir, J. Shalf, D.H. Bailey, L. Oliker, M. Banda, C. W. McCurdy, J. Hules, A. Canning, M. Day, P. Colella, D. Serafini, M.F. Wehner, and P. Nugent. Science-driven system architecture: A new process for leadership class computing. *Journal of the Earth Simulator*, 2, January 2005.

[60] D. Skinner. Integrated Performance Monitoring: A portable profiling infrastructure for parallel applications. In *Proc. ISC2005: International Supercomputing Conference*, volume to appear, Heidelberg, Germany, 2005.

[61] E. Strohmaier and H. Shan. Architecture independent performance characterization and benchmarking for scientific applications. In *International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, Volendam, Netherlands, Oct 2004.

[62] S. Succi. The lattice boltzmann equation for fluids and beyond. *Oxford Science Publ.*, 2001.

[63] Jeffery S. Vetter and Frank Mueller. Communication characteristics of large-scale scientific applications for contemporary cluster architectures. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS)*, 2002.

[64] Jeffrey S. Vetter and Andy Yoo. An empirical performance evaluation of scalable scientific applications. In *Proceedings of the IEEE Conference on Supercomputing*, 2002.

[65] J. White. An optimization experiment with the community land model on the Cray X1. In *Proc. of the 45th Cray Users Group Conference*, Columbus, OH, 2003.

[66] Y-G Yoon, B.G. Pfrommer, S.G. Louie, and A. Canning. NMR chemical shifts in amino acids: effects of environments, electric field and amine group rotation. *Solid State Communications*, 131, 2004.